**ID: Crawler-1**

**Priority:** 10

**Cost:** 20

**Name:** Basic Web Crawler

**Description:** The system should be able to visit all of the pages on the website.

**Precondition:** **Report-1 (**A scan has been initiated )

**Primary flow:**

1. The web crawler will initialize a visited list and a pending queue.

2. The web crawler will retrieve the root page of the domain.

3. The web crawler will parse the retrieved page for resource URLs.

4. The web crawler will create a page object with the URL, a null parent URL, and a list of the child resource URLS that were retrieved.

5. The web crawler will place this page object in the pending queue.

6. If there is a page object in the pending queue, it will be popped, and become the current page. If there is not, the scan is finished, and the crawler exits.

7. The scanner will iterate over all of the child resources in the current page object.

8. If the current child resource is a page object in the visited list or the pending queue, continue with 7.

9. The scanner will create a page object using the current page as the parent.

10. The current child resource will be retrieved.

11. If a page is retrieved, resources will be parsed from the page and appended to the child resource list of the new page object.

12. The new page object will be added to the pending queue.

13. Return to 6.

**Alternate flow:**

1. The web crawler will initialize a visited list and a pending queue.

2. The web crawler will retrieve the root page of the domain.

3. If the root page cannot be retrieved, the scan exits.

**Postconditions:** Every URL referenced in any page reachable from the root page will be visited. This does NOT guarantee that it will be retrieved, as it may not exist.

**ID: Crawler-2**

**Priority :** 10

**Cost:** 20

**Name:** Record Crawler Results

**Description:** The results of crawling the website will be recorded in a database so that they can be used later.

**Preconditions:** The webcrawler must be running. The webcrawler must have generated a record in the "scans" table and retrieved a scan ID.

**Primary flow:**

1. When a page object is created, a matching entry in the database will be created, including the scan id, a link to the parent page (this can be null for the root of the scan), the current URL, the retrieval time, and the HTTP response code.

2. The resource\_children table will be updated to create a link between the parent and the new child object.

3. The resource id of the newly created record will then be stored in the page object.

**Alternate flow:**

1. When a child resource is found to already have an associated page object in the pending or visited queue, a link will be created in the resource\_children table between the current page object and the one that was found.

**Postconditions:** The resources table will contain a complete list of all reachable nodes on a graph of the website. The resource\_children table will contain a complete list of the reachable edges of a graph of the website. The resources table will also contain results codes, retrieval times and a spanning tree of the graph of the website. As the response codes will be recorded and records will be created even for pages that are not accessible, this will be a record of broken links as well as working ones.

**ID: Report-1**

**Priority :** 10

**Cost:** 20

**Name:** Crawler Interaction

**Description:** The user must be able to start new scans and list the completed scans.

**Preconditions: ID-13** (Login)

**Primary flow:**

1. The system will display a button to start a new scan.

2. The system will retrieve a list of scans that have been completed.

3. The system will display the list of completed scans.

4. The user may click on the "Start Scan" button.

5. When the button is clicked, a scan will be dispatched via a call to exec();

**Alternate flow:**

1. The system will display a button to start a new scan.

2. If there are no scans completed, none will be listed.

3. The user may click on the "Start Scan" button.

4. When the button is clicked, a scan will be dispatched via a call to exec();

**Postconditions:** The user will know what scans have been completed. If the button was depressed, a scan will have been started.

**ID: Report-2**

**Priority :** 10

**Cost:** 30

**Name:** Show Scan Results

**Description:** As a user, I must be able to view reports of the scans, showing all irretrievable resources under my domain. I should be able to do this from a website.

**Preconditions: Crawler-1, Crawler-2** (A scan must have been recorded), **Report-1** ( the user must have a list of scans)

**Primary flow:**

1. The user will select a scan that has been run.

2. The scan container will open.

3. The scan container will request the scan data from the webserver.

4. The webserver will retrieve all page resources that match this scan id, format them as JSON, and return them to the scan container.

5. The scan container will then display the JSON results.

**Alternate flow:** None

**Postconditions:** The scan will be displayed.

**ID: Report 3**

**Description:** The system will be able retrieve the data to view scans and start new scans from a website.

**Precondition: Crawler 1,Crawler 2,Report 1, Report 2**

**Primary Flow:**

1) The system will initiate a scan by when the scan button pressed in the website's UI.

2) When the scan is complete, the system will retrieve the results from the scan display results in the UI.

**Post condition:** All the data from the scan will be pulled from the system's database and displayed in the website's UI.

**ID: Report 4**

**Description:**  The system will be able to sort the reports that are presented to me and sort them by the order in which the pages were visited, alphabetically, and alphabetically by the parent page.

**Precondition: Crawler 1,Crawler 2,Report 1, Report 2, Report 3**

**Primary Flow:**

1) The system will retrieve data from the database.

2) The system will sort data based on the sorting order selected by the user.

3) The system will display the results of the sort in the website's UI.

**Postcondition:** The system will display reports sorted in the manner selected by the user.

**ID: Report 5**

**Description:**  As a user viewing a report, I should be able to generate that report in a printer friendly format.

**Precondition: Crawler 1,Crawler 2,Report 1, Report 2, Report 3, Report 4**

**Primary Flow:**

1) The system will retrieve data from the database.

2) The system will sort retrieved data.

3) The system will display sorted or unsorted data in the website UI.

4) The system will parse the data generated in the web UI.

5) The system will create a printable text based format of the data displayed in the website's UI.

6) The system will display the printable version of the scan results in a new window.

**Postcondition:** The system will generate a new window displaying a printable text based format of the results of the scan.

**ID: Report 6**

**Descripton:** The system will be able to select two scans and show only the items that have changed.

**Precondition: Crawler 1,Crawler 2,Report 1, Report 2, Report 3, Report 4**

**Primary Flow:**

1) The system will be able to retrieve the results of two scans.

2) The system will be able to compare the results of the two scans.

3) The system will be able to display the results of the two selected scans that do not match.

**Postcondition:** The system will be able to display the differences between two scans.

**ID: Report 7**

**Description:**  The system will be able to display reports from scans that are in progress.

**Primary Flow:**

1) A scan will be started by the system.

2) The system will be able to retrieve data from the database while a scan is in progress.

3) The system will be able to display data from the database in the web UI while a scan is in progress.

**Postcondition:** The system will be able to retrieve data from the database while a scan is in progress.

**ID:** UC 9  
**Priority:** 6 **Cost:** 12  
**Name:** Runtime Limit  
  
**Description**: The scan can take a long time to fully run. As a result, this use case is designed to allow a user to limit the run span of the scan. The user can decide to have the scan run as much as it can in a certain amount of time, or to limit the scan to only run until it gets to a certain distance away from the main page.   
  
**Pre-Condition**: All features of the scan are completed including searching through all the pages, reporting the errors, pausing, and stopping.  
  
**Standard Flow**:  
1. The user decides how he/she wants to limit the search (either length or depth) and enters the appropriate value in the appropriate location for the choice. This starts the scan.  
2. The scan will run as normal and go through all of the pages until it reaches the limit set by the user.   
3. The finished report with all the errors for the pages searched will be displayed.  
  
**Alternate** **Flow 1**:  
1. The first step will be the same as the standard flow.   
2. The scan will run until the user pauses the search. This will also pause the timer if the user chooses to do a time based search. The scan will remained pause until the user resumes the search.   
3. Same as step 4 in the standard flow.

**Alternate Flow 2**:  
1. The first step will be the same as the standard flow.   
2. The scan will run until the user stops the search. This will terminate the search.   
3. The final report will be printed and displayed all of the errors up to the termination of the search, which will not be the same as what the user initially requested.   
  
**Alternate Flow** **3:**  
1. The user enters invalid information for one of the two limiting factors trying to be used.   
2. An error message will be displayed informing the user that the information entered is incorrect and prompt them to try again.   
3. Return to step number 2 in the standard flow.

**Post Conditions**: A scan will be completed based on the limiting factor given by the user. A report will be completed and displayed for the user to examine. A record of the scan will be saved in the database so that it can be accessed in the future, used to be compared to other searches, or used to start another search to check for errors being fixed.

**ID:** UC10  
**Priority:** 5 **Cost:** 10  
**Name:** Error Check  
  
**Description:** This use case will select a scan that has been run previously. A second scan will begin and will only rescan what was scanned in the previous results that were initially selected. It will return a report and show a comparison of the two reports to show what errors are the same, what errors are new, and what errors are fixed (if anything has been).  
  
**Pre-Conditions:** All scanning and reporting features are working properly including searching the SPSU domain, reporting errors, displaying errors, pausing scans, stopping scans, and comparing two reports.  
  
**Standard Flow:**  
1. The user will select a previous report that he/she wants to see if any changes were made.   
2. The user will start a comparative scan.  
3.The scan will run and scan the same pages as the previous scan.   
4. A report will be generated based on the scan that was just run.   
5. The two scans will be compared, showing what errors were still present, any new errors that popped up, and showing which errors were fixed.  
  
**Alternate Flow 1:**  
1.The first 3 steps from the standard flow will be followed.   
2. The user will pause the scan during the search.   
3. A partial report will be shown, displaying what errors have been encountered thus far.   
4. The scan remains paused until the user resumes the scan. The scan continues on normally.   
5. Return to step 5 in the standard flow.   
  
**Alternate Flow 2:**  
1.The first 3 steps from the standard flow will be followed.   
2. The user will stop the scan during the search. This terminated search.   
3. A report will be created based on the partial scan, and the report will compare only the parts of the scan that were completed.  
  
**Post Condition:** A repeat scan will have run. There will be two reports based off of the same scan. They will be compared and the user will see what. If anything, has changed for either better or worse.  
  
  
 **ID:** UC11 **Priority:** 5 **Cost:** 8 **Name:** Secure Check **Description:** As the web crawler goes through the SPSU domain, it will come across links that are secure. These items should be listed as secure login links and not as errors (as they can not be accessed) and shown as such on the report that is displayed at the end.  **Pre-Conditions:** Web crawler, error reporter, and report viewing is working properly.  **Standard Flow:**1. User starts a scan.   
2. Web crawler and error reporter are called and begin to work.   
3. As the crawler tries to access something in the SPSU domain that requires a secure login, it will report that to the database as a link to a secure site.   
4. When the scan has completed, the report will populate all of the items from the scan. The secure links will appear on the report, and will be listed as secure links instead of errors.  **Post Conditions:** The user will have an updated report that no longer lists the secure pages as a link that can’t be accessed, but as a secure page that requires a login.  **ID:** UC12  
**Priority:** 3 **Cost:** 14 **Name:** Subdomain Sort **Description:** When looking at the report, we will be able to sort it based on the subdomain. This will allow us to break down the report and look at the errors only in certain subdomains, which would be useful to do something along the lines of finding all of the errors in a certain department’s webpages.  **Pre-Conditions:** Web crawler, error reporter, and report viewing is working properly.  **Standard Flow:**1. User starts a scan and receives a report, or opens a report that has been previous run and recorded.   
2. User clicks on the sort feature that says subdomain.   
3. A sorting algorithm (using pattern matching) is used to arrange the items in the report based off of their subdomain. The default case will be alphabetical. If the button is pressed twice, the report will be in reverse alphabetical order.  **Post Conditions:** The user has a viewable report that properly lists the items based on their subdomain either from a-z or from z-a.   
  
**ID:13**

**Name:** Login

**Priority:** 3

**Cost:** 2

**Description:** This is a security feature that prevents unauthorized users form making the product into a Denial of Service attack platform or find information about the target website that might be exploitable. This login will be set on creation of the system. The login will require a user name and simple password. The login information will be stored in the data base.

**Preconditions: Report-2**

**Standard flow:** User will be prompted for user name and password upon arrival to the website. The user will then type in a user name and password into there appropriate blanks and press the login button or enter.

**Alternative flow:** IF the user does not enter a field such as password or user name the login will fail. If user enters a wrong user name or password the login will throw a invalid login error. Should the user try and bypass this step web pages beyond this will not be displayed.

**Post Conditions:** After a successful login the user will be able to go beyond the login page and view or start scans.

**Considerations or issues:** This is a low priory by the customer however for security reasons needs to be implemented. There also is no registration now and requires the system admin to add new login entries.

**ID:14**

**Name:** Pause Scan

**Priority:** 2

**Cost:** 4

**Description:** This will temporally pause the web crawler aspect of the scan. This will not stop processes such as report builders. This option is only available after a scan has started. Once the scan has been stopped the user can continue scan at the press of a button.

**Precondition: Crawler-1**

**Standard Flow:** User has started a scan and then presses the pause scan. The scan stops and can be restart by pressing continue scan.

**Alternative Flow:** Scan is unable to restart so error is thrown. User is able to access stop scan before scan has started do nothings.

**Post Conditions:** Scan continues.

**Considerations/ Issues:** This might include pausing threads.

**ID:** 15

**Name:** Timer

**Priority:** 1

**Cost:** 6

**Description:** This allows a user to run a scan at a user set time. This involves the server knowing what time it is and starting a process based on that. The scan will not display a live report unless a user has logged in during its run. This can be set up to run once, weekly or monthly.

**Precondition:** **Crawler-1, Crawler-2, Report-1, Report-2,8**

**Standard Flow:** A user has logged on and imputed a time and date in the required box and check the required boxes dealing with now often this reoccurs. Should no options be checked and no date and only a time the default is run once.

**Alternative Flow:** Should the user input nothing no scans will be scheduled. Should an impossible time be imputed no scans will run and a message will display to the user that this is an invalid time.

**Post Conditions:** A scan will run

**Considerations/ Issues:** This can be used to make a Denial of service attack if abused. However this scan could be slowed down should this become an issue.

**ID: 16**

**Name:** Stopwatch

**Priority:** 1

**Cost:** 2

**Description:** This will add into the report the time it took to load a page.

**Precondition: Crawler-1**

**Standard Flow:** During the web crawler's execution times will be recoded of the load times of web pages.

**Alternative Flow:** None

**Post Conditions:** There is now a field in the data base that holds web load times

**Considerations/ Issues:** Adds a new field to the data base which could potential slow down the scan.